



**Seventh Annual  
AUUG Canberra  
Summer Conference**

**ANU  
February 14, 1996**



samba  
novell server/client

dir ./ or dir ..\n  
connect IPC:ICP

## CONFERENCE PROGRAM

Lin/ux User Group

4th Thu.

9:00	Welcome
	<i>Merik Karman, AUUG Canberra President</i>
9:10	Samba <a href="http://samba.canberra.edu.au/pub/samba">http://samba.canberra.edu.au/pub/samba</a>
	<i>Andrew Tridgell, ANU</i>
9:45	IPv6 - The Next Generation
	<i>Peter Elford, Cisco Australia</i>
10:20	Morning Tea
11:00	Government Publishing on the Web: An Example Management Tool for Large Data Repositories
	<i>Bruce McLeod, Approved Systems</i>
11:35	Monitoring Network Connection Attempts on a FreeBSD Server
	<i>Warren Toomey, ADFA</i>
12:10	Lunch
1:30	Aegis is Only For Software, Isn't It?
	<i>Peter Miller, AGSO</i>
2:05	Plan 9
	<i>Chris Vance, ADFA</i>
2:40	Afternoon Tea
3:20	Linux Update
	<i>Stephen Rothwell, NEC Information Systems Australia</i>
3:55	The PDP Unix Preservation Society
	<i>Warren Toomey, ADFA</i>
4:30	Close



## **CONFERENCE ABSTRACTS**

The Conference was held on Wednesday 14th February in ANU at 9am. The program included the following presentations:

### **Samba**

*Andrew Tridgell, ANU*

Samba is a free SMB fileserver for Unix. It allows a Unix box to act as a file/print server for PCs which use the very common SMB protocol. Samba is in use at thousands of sites worldwide. In this talk I'll give an overview of Samba. I'll give a few technical details, design criterion etc, as well as a few anecdotes. I'll also give a bit of a description of what I think the future of Samba will be.

### **IPv6 - The Next Generation**

*Peter Elford, Cisco Australia*

Why does there need to be an IPv6? How is it different from whatever version of IP we are using now? This talk provides some background to the requirements for and development of the IPng (Next Generation) protocol. A brief overview of the features offered by IPv6 will also be provided.

### **Government Publishing on the Web: An Example Management Tool for Large Data Repositories**

*Bruce McLeod, Approved Systems*

This paper discusses some of the data management issues concerned with the storage of large data repositories of text. Normally in Government there are large existing repositories of data either present as legacy data on mainframes or as file servers full of word processing documents. Without having to go through a large HTML conversion process and data redundancy headache, the development documented in this paper describes how data can be: free text searched, dynamically formatted as HTML, and presented to the user so that documents can be delivered by: FTP, EMAIL, or direct FAX.

### **Monitoring Network Connection Attempts on a FreeBSD Server**

*Warren Toomey, ADFA*

Tools such as Strobe and SATAN allow both network administrators and hackers to test vulnerable points in a networked system's security. This presentation looks at kernel modifications to a FreeBSD server which monitor and log these sorts of activities. A summary and review of the last six months of log information will be given for the well-known ftp/WWW server minnie.

## **Aegis is Only For Software, Isn't It?**

*Peter Miller, AGSO*

Aegis is a Software Configuration Management system, which provides a method for managing concurrent development and peer review with strong auditability. Other systems being managed with Aegis at AGSO are DNS and the Web. Using Aegis to manage DNS provides a reliable way to maintain and check DNS tables. The system is peer reviewed, so no 'broken' changes are able to get into the system tables. Using Aegis to manage the AGSO Web models the production of scientific papers. In the normal publication process an author writes a paper and then it is peer reviewed, the reviewers may return it with comments or approve it to the publisher. The publisher in turn may accept it for publication or return it. A similar model is available using Aegis when publishing Web pages; the publication analogy is deliberate since the work is indeed available to the public. The 'build' step is used to resolve server-side includes, check HTML and to generate various indices. Some Aegis reports are also used, such as the one from which the 'What's New' page is generated. The provision by Aegis of individual 'sand pits' greatly facilitates concurrent development of Web pages and improves productivity.

## **Plan 9**

*Chris Vance, ADFA*

Plan 9 from Bell labs is the latest in research operating systems from the organisation which gave us Unix. Plan 9 is a distributed system. In the most general configuration, it uses three kinds of components: terminals that sit on users' desks, file servers that store permanent data, and other servers that provide faster CPUs, user authentication, and network gateways. These components are connected by various kinds of networks. Plan 9 is now available on a PC platform for experimentation and evaluation as will be demonstrated.

## **Linux Update**

*Stephen Rothwell, NEC Information Systems Australia*

Linux is a freely available Unix-like operating system for several different hardware platforms. This talk will present a summary of where Linux came from, an overview of where Linux is now and an (educated) guess of where it is going. I will also attempt to give a rationalisation of why we use (and enjoy) Linux.

## **The PDP Unix Preservation Society**

*Warren Toomey, ADFA*

Unix is now over 25 years old, having been started by Ken Thompson and several others at Bell Laboratories in 1969. For most of the 70's, Unix was developed and used on PDP-11s with 256K of memory or less, and supporting dozens of users. A PDP Unix Preservation Society has been started to not only to preserve the source code, binaries, anecdotes, urban folklore of the era of Unix on PDPs, but also to form a 'user group' of people who are still interested in PDP Unix and/or who are still running Unix on PDPs, so as to share experiences, ideas, tips and answers about this old software. The presentation will look at the features of the PDP Unixes, give some folklore, and hopefully demonstrate Sixth or Seventh Edition Unix in action.

# IPng: The Next Generation Internet Protocol

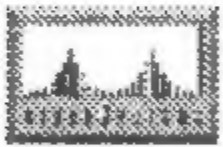


Peter Elford  
Senior System Engineer  
Cisco Systems Australia  
pelford@cisco.com



## Agenda

- Key Issues
- IPng History
- IPng Alternatives
- IPng Overview
- Selected Features
- Transition
- Discussion



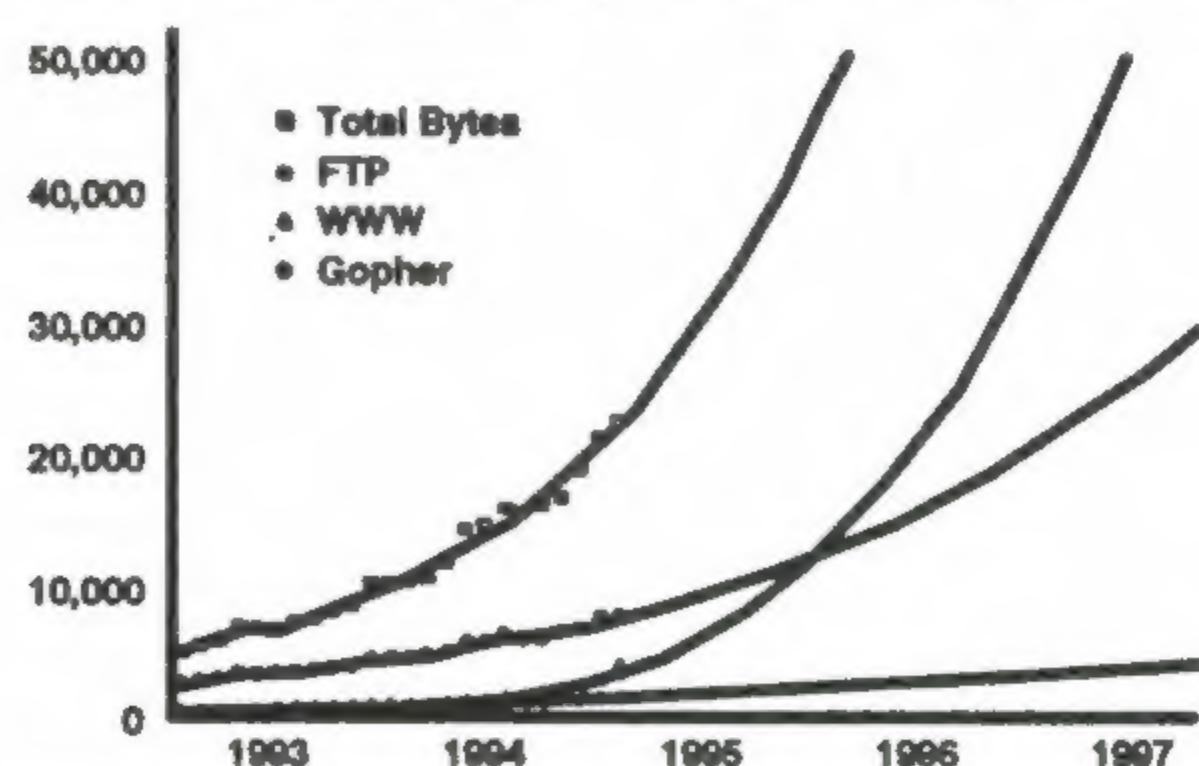
## Motivation

- Several key issues facing existing IP
- Growth  
Internet, PDA's, cable TV, toasters, ...
- New Functionality  
Mobile hosts, Multimedia, ATM, ...
- Transition  
God created the world in 7 days because  
there was no installed base



## Internet Growth

Bytes Transferred per Month (in Billions)



\* Source: <http://www.cc.gatech.edu/gvu/state/NSF/Extrap.GIF>



## IPv4 Address Scaling Problems

- Class C too small; everyone getting B's  
1990 prediction: B's gone by March, 1994
- Multiple class C addresses increases  
routing table size  
Global routing table growing 1.5x faster than memory
- IPv4 address space will run out  
Although 4 billion in theory, allocation is wasteful



## ► Short Term Fix: CIDR Classless InterDomain Routing

- Contiguous route entries grouped

203.6.40.0	255.255.255.0	
203.6.41.0	255.255.255.0	
203.6.42.0	255.255.255.0	
203.6.43.0	255.255.255.0	("a /24 prefix")

Becomes

203.6.40.0	255.255.252.0	("a /22 prefix")
------------	---------------	------------------



## ► Other Options

- Private Address Spaces  
RFC1597
- Application Layer Gateways  
Proxy Servers
- Network Address Translation



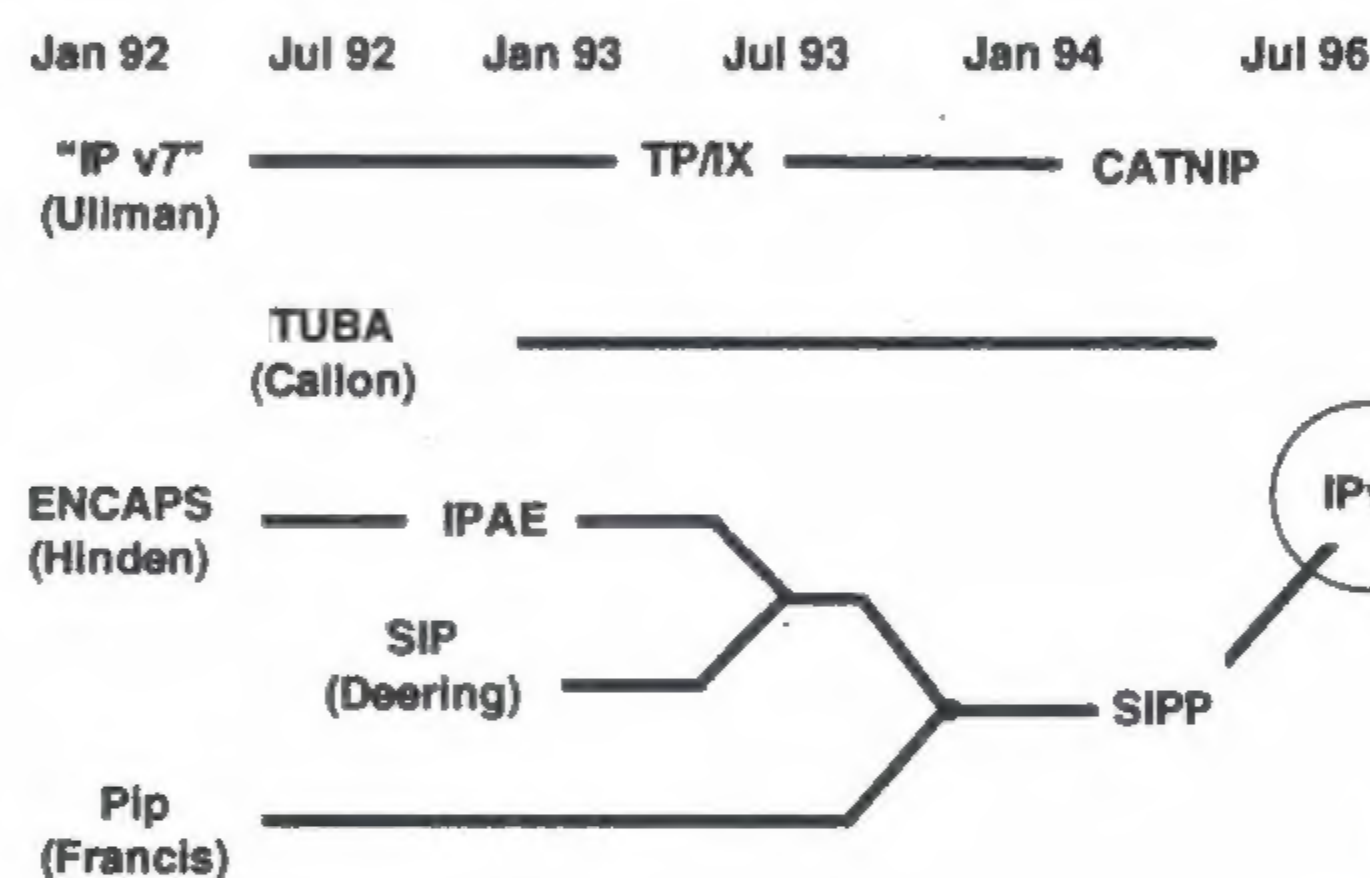
## ► Long Term Solution

- Routing problem
- Address space problem

- 2.5 Year Effort by IETF to define the Next Generation IP



## ► IPng Candidates



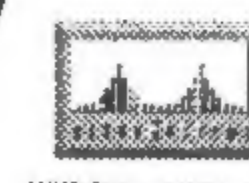
## ► Why IPv6 ?

- 0 - 3 unassigned
- 4 Internet Protocol, IP (current version)
- 5 Stream Protocol, ST (not an IPng)
- 6 SIP - SIPP - IPv6
- 7 IPv7 - TP/IX - CATNIP
- 8 PIP
- 9 TUBA
- 10-15 unassigned



## ► IPv6 Features

- Expanded Addressing
- Streamlined Header
- Improved support for Options
- Quality of Service Capabilities
- Authentication and Privacy
- Mobility, Autoconfiguration, Security



## ► Header Format

Vers	Prio	Flow Label	
Length		Next HDR	Hop Limit
Source Address			
Destination Address			



AUXO Cardano 1998 13

## ► Changes from IPv4

- (MUCH) bigger addresses
- Fragmentation and options separate headers
- Header Checksum and Length eliminated
- TOS eliminated
- Added Flow label
- TTL rename (correctly) hopcount
- Protocol renamed Next Header
- Precedence renamed Priority



AUXO Cardano 1998 14

## ► Addressing Syntax

- Preferred Form  
1080:0:FF:0:8:800:200C:417A
- Compressed Form  
FF01:0:0:0:0:0:43 becomes FF01::43
- IPv4 Compatible  
0:0:0:0:0:0:13.1.68.3 (or ::13.1.68.3)



AUXO Cardano 1998 15

## ► Address Types

- Unicast (one to one)
  - global, link-local, site-local
  - compatible (IPv4, IPX, NSAP)
- Multicast (one to many)
- Anycast (one to nearest)



AUXO Cardano 1998 16

## ► Routing

- Hierarchical Addresses
- Distributed address registries
- New versions of existing Routing Protocols
- No change ...



AUXO Cardano 1998 17

## ► Flows and Autoconfiguration

- Flows  
Sequence of packets with "real-time" needs  
Explicit setup, packets labelled with flow ID
- Autoconfiguration  
Stateful (like DHCP but simpler)  
Stateless (create address from prefix)



AUXO Cardano 1998 18

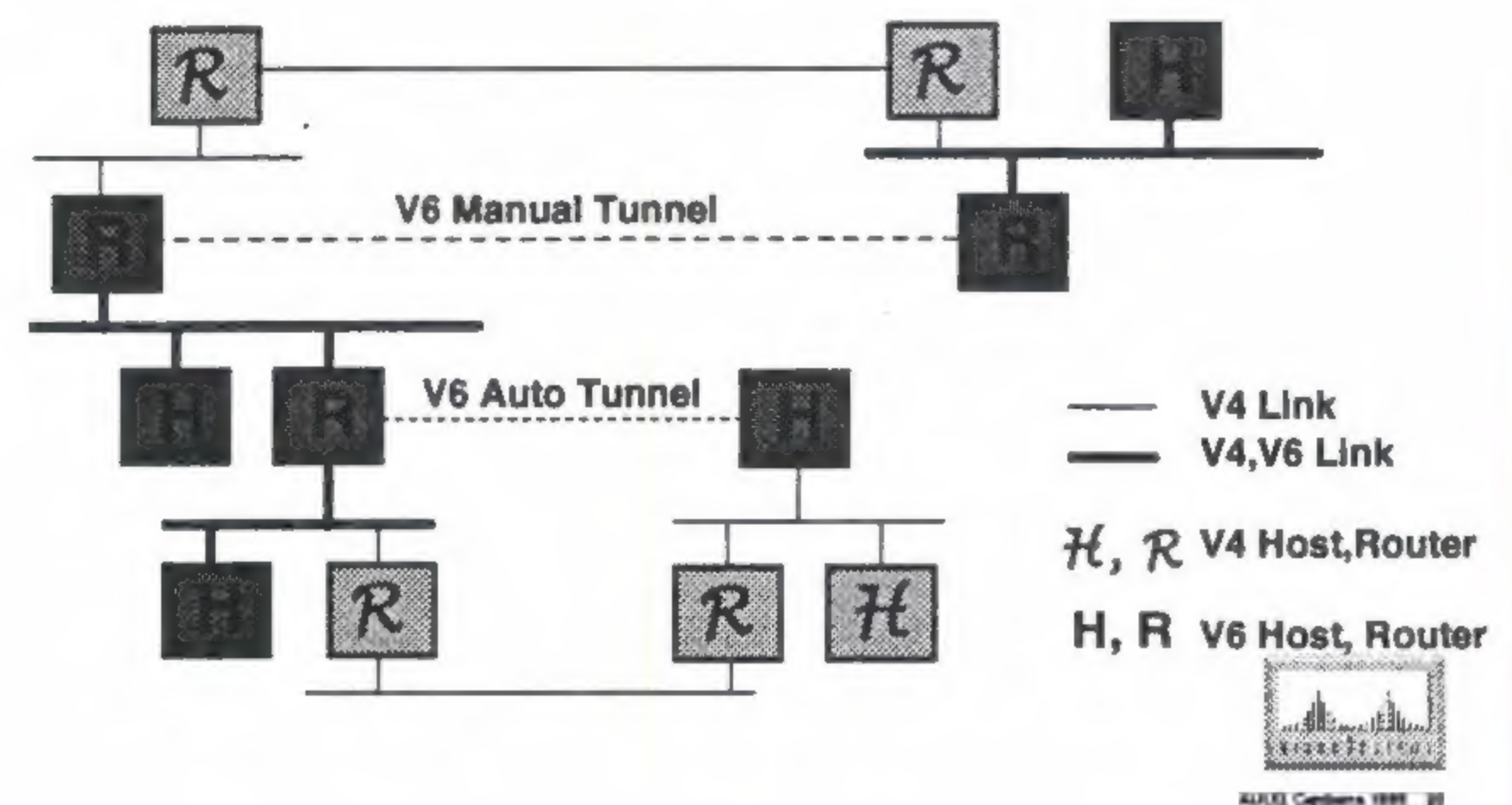
## ► Transition

- Upgrade IPv4 hosts and routers to also support IPv6
- Link islands of IPv6 with "tunnels"  
IPv6 travels in IPv4 packets between islands
- Slowly migrate away from tunnels as IPv6 expands
- Host applications rewritten
- New DNS type (AAAA) - IPv6 address



ALIX Carbone 1999 19

## ► Transition Components



ALIX Carbone 1999 20

## ► So Do I Care ?

- Right now - Probably Not
- Deployment will take time
- Networks exist to run applications!!!
- Bake off early first quarter
- Market Acceptance ...
- More ...

RFC 1752

<http://playground.sun.com/pub/png/htm;/png-main.htm>



ALIX Carbone 1999 21

# Monitoring Network Connection Attempts on a FreeBSD Server

Warren Toomey  
wkt@cs.adfa.oz.au

School of Computer Science  
Australian Defence Force Academy

## Abstract

*Tools such as Strobe and SATAN allow both network administrators and potential intruders to test vulnerable points in a networked system's security. This presentation looks at daemon programs and kernel modifications to a FreeBSD server which monitor and log these sorts of activities. A summary and review of the last six months of log information will be given for the FTP and WWW server minnie.cs.adfa.oz.au.*

## Introduction and Motivation

The growth of the Internet brings with it a higher risk of server attack by would-be intruders. Rather than saying that the risk of attack is now higher, it would be truer to say that these days you will be attacked sooner rather than later. The methods that network intruders use to attack systems are also becoming more and more sophisticated. They are now using defects in the IP and TCP protocols themselves (and their implementations) to penetrate machines connected to the Internet.

There are many tools available for network administrators to discover possible weaknesses in a server's network security; SATAN and Strobe are two examples. These tools are just as useful to would-be intruders. 'Underground' FTP and WWW sites also have tools to perform attacks such as IP spoofing, shared-library attacks, and to exploit known holes in such programs as `sendmail` and `telnetd`. Existing operating systems do not detect the use of these tools and methods, nor do the systems let the network administrator know of their use.

Over the past few years I have been running a small FTP and WWW site to provide information about FreeBSD and NetBSD. I've also been involved in setting up an Internet firewall at ADFA. After reading several papers and books about network security and firewalls, I quickly realised how vulnerable most network systems are. As part of the security tightening on my FTP server, I was motivated to modify the source code of the server's kernel to detect attempts to connect on unused network ports, and to reject packets with 'suspicious' options. These modifications allow me to observe many of the attacks described above and may also highlight those services which intruders believe to be vulnerable, even before CERT or AUSCERT finds these vulnerabilities.

Recently I started to construct other kernel modifications to prevent the IP and TCP spoofing attacks mentioned and to log the incoming packets for services such as TFTP, NFS and RSH. Although none of these have been triggered yet, they should provide more information about the intentions of a would-be intruder.

## Server Description

I run minnie.cs.adfa.oz.au, a FreeBSD 1.1 network server with the following services:

- Anonymous ftp to 500M of files, using WU-FTP 2.4,
- Web service to 100M of files, using Apache 0.8.8,
- Email using Sendmail 8.6.10,
- NTP using Xntpd 3.3g,

- Telnet service with Berkeley Telnetd,
- Rsh and rlogin services (only on my subnet) using Berkeley Rshd and Rlogind,
- Converse service, a talk-like protocol used in amateur radio, and
- Domain Name service using Berkeley Named 4.8.3.

Minnie runs no other services. I use the wonderful tool called TCP\_Wrappers, written by Wietse Venema, to protect services such as rsh, rlogin and converse. TCP\_Wrappers allows or denies connections to particular services depending on the IP address where the request comes from. On Minnie, requests to these services from outside my department are rejected, and the requests are logged. TCP\_Wrappers is a great utility, and if you're running any Internet connected Unix machine, I'd recommend using TCP\_Wrappers on it.

## Kernel Modifications

Minnie is a small system, but its security is still important, at least to me! I've made lots of changes at the user-level to improve security, but these changes won't protect the system from attacks at the IP and TCP level. As full kernel sources for FreeBSD are available, it makes sense to add code to the kernel to log suspicious network behaviour. Any modifications made to the kernel must be very well scrutinised to ensure that they do not introduce new vulnerabilities to the system.

The modifications I've made are given below as diffs against the FreeBSD 2.1 kernel. Unless otherwise noted, these should be applicable to nearly all systems with network stacks derived from 4BSD. As with any kernel modifications, don't apply them blindly; read and understand them first. I'd appreciate comments about the changes from people who have a lot of experience with the BSD network code.

## Monitoring Unused Ports

The FreeBSD kernel was modified to log TCP connection packets for ports which are not in use (tcp\_input.c):

```
*****
*** 376,381 ****
--- 378,391 ----
        * but should either do a listen or a connect soon.
        */
        if (inp == NULL) {
+ #ifdef LOG_TCP_BADPORT
+         /* Log connection attempt */
+         if (tiflags & TH_SYN) {
+             log(LOG_INFO, "Conn attempt on TCP port %d from %x port %d\n",
+                 ntohs(ti->ti_dport), ntohl(ti->ti_src.s_addr),
+                 ntohs(ti->ti_sport));
+         }
+ #endif
            goto dropwithreset;
        }
        tp = intotcpcb(inp);
        if (tp == 0)
```

The kernel was also modified to log UDP data packets for ports which are not in use (udp\_usrreq.c):

```
*****
*** 268,273 ****
--- 268,279 ----
                                uh->uh_dport, INPLOOKUP_WILDCARD);
        }
        if (inp == NULL) {
+ #ifdef LOG_UDP_BADPORT
+         /* Log connection attempt */
+         log(LOG_INFO, "Conn attempt on UDP port %d from %x port %d\n",
+           ntohs(uh->uh_dport), ntohl(ip->ip_src.s_addr),
+           ntohs(uh->uh_sport));
+ #endif
        udpstat.udps_noport++;
        if (m->m_flags & (M_BCAST | M_MCAST)) {
            udpstat.udps_noportbcast++;
        }
    }
}
```

These changes don't log the contents of the packets for two reasons. Firstly, logging packet contents will slow down the kernel and lower the system's performance. Secondly, the amount of data logged will be enormous; Minnie receives broadcast packets to UDP port 520 every few minutes from the subnet's router, not something that I'm particularly interested in logging. I will discuss a user-mode solution to packet logging later.

## Logging Root Execs

I also modified the FreeBSD kernel to log all `exec()` system calls where the effective user-id was root (kern\_execve.c). This patch is specific to FreeBSD 1.1 and won't automatically apply to other kernels.

```
*****
*** 286,291 ****
--- 286,304 ----
        p->p_cred->p_svuid = p->p_ucred->cr_uid;
        p->p_cred->p_svgid = p->p_ucred->cr_gid;

+ #ifdef LOG_ROOT_EXECS
+     /* Print out exec information for processes running as root */
+     if (p->p_cred->pc_ucred->cr_uid == 0) {
+         log(LOG_NOTICE, "ROOT exec ");
+         for (i=0, stringp=iparams->stringbase; i<20 && i<iparams->argc; i++) {
+             log(LOG_NOTICE, "%s ", stringp);
+             /* Walk stringp up to next string */
+             while (*stringp++);
+         }
+         log(LOG_NOTICE, "\n");
+     }
+ #endif
+
+     /* mark vnode pure text */
+     ndp->ni_vp->v_flag |= VTEXT;
```

This won't help stop a successful attack on Minnie, but will at least provide an audit trail of commands that the intruder performed as root. To date, I have seen nothing in the logs that could not be accounted for by normal root activity.

## Reading the Logs

There's no point in logging suspicious activity if the logs are not read. Reading the raw log files is tedious and mind-numbing; they do need to be kept to provide an audit trail in case you are broken into. I have the last three years of log files compressed and archived in multiple places. As well, the raw log files are written to Minnie and another machine by `syslogd`, to minimise the impact if an intruder gets in and alters or destroys the logs on Minnie.

All system administrators need a tool which summarises the log files and highlights what the system administrator considers suspicious activity. How frequently the tool runs and what it highlights depends on the system's security requirements. On Minnie a cron job reads through the logs daily, looking for suspicious behaviour. The result is emailed to me on another machine. A part of the summary shows TCP\_Wrapper's activity and the results of my kernel modifications. Here is an example report:

From: Minnie Bannister  
Date: Fri, 26 Jan 1996  
To: wkt@cs.adfa.oz.au

### TCP Connections

```
-----
ftp connections:      378      ftp refusals:      0
sendmail connections: 10      sendmail refusals:  0
finger connections:   0       finger refusals:    2
rlogin connections:   0       rlogin refusals:    0
rsh connections:      0       rsh refusals:       0
telnet connections:   2       telnet refusals:    0
```

```
08:53:49 fingerd[13074]: ccadfa.cc.adfa.oz.au
10:28:07 fingerd[18732]: rudolph.north.pole.com
```

### Attempted TCP Port Connects: 1

```
-----
Port 1713: 1 attempt
```

```
14:21:48 TCP proto 1713 from 206.20.189.10, port 1075
```

### Attempted UDP Port Connects: 3

```
-----
Port 33492: 1 attempts
Port 33493: 1 attempts
Port 33494: 1 attempts
```

```
10:50:12 UDP proto 33492 from 129.72.40.4, port 41776
10:50:13 UDP proto 33493 from 129.72.40.4, port 41776
10:50:14 UDP proto 33494 from 129.72.40.4, port 41776
```

The first section summarises the information from TCP\_Wrappers. There were only two connection refusals, both to the `finger` service. The next two sections show the attempted connects to unused TCP and UDP ports. The UDP connections show a `traceroute` to Minnie. I have no idea what's on TCP port 1713; the program which produced the summary would have given the name of the port if it was in the file `/etc/services`.

## Six Months of Refused Connections

Let's look at some of the refused connection statistics for the last six months. Data from my local subnet has been omitted to remove 'noise' caused by broadcast packets. There were very few suspicious connection attempts from within ADFA.

### TCP Port Attempts for June - December 1995

Port	Service	Jun	Jul	Aug	Sep	Oct	Nov	Dec	Total
113	ident	309	469	513	509	391	973	875	4039
119	nnntp	17	16	14	22	9	111	13	202
70	gopher	13	4	5	3	6	17	4	52
3105		0	1	0	0	0	36	0	37
3767		0	32	0	0	0	0	0	32
1343		29	0	0	0	0	0	0	29
4669		0	27	1	0	0	0	0	28
1969		0	0	0	0	19	0	0	19
8001	alt-www	0	0	4	0	2	12	1	19
1254		0	0	0	0	0	15	0	15
2072		0	0	14	0	1	0	0	15
4047		1	0	12	0	0	0	0	13
87	ttylink	1	3	4	2	2	0	0	12
2000	callbook	0	0	1	1	8	0	0	10
4154		0	0	0	0	0	10	0	10
8080	alt-www	0	0	4	1	0	2	3	10
0		7	1	0	1	0	0	0	9
3029		0	6	0	2	0	0	0	8
3955		0	0	2	0	0	0	6	8
1156		0	0	7	0	0	0	0	7
1859		5	0	1	1	0	0	0	7
1903		0	3	2	0	2	0	0	7
2950		0	0	5	2	0	0	0	7
3179		0	0	6	1	0	0	0	7
513	rlogin	2	1	1	0	1	0	1	6
1095		4	1	0	0	0	1	0	6
1538	3ds-lm	0	5	1	0	0	0	0	6
2012	ttyinfo	0	0	1	4	0	1	0	6
131	cisco-tna	3	0	0	0	0	0	2	5
2016	bootserver	0	0	0	3	0	0	0	3
1993	snmp-tcp-port	0	0	1	0	0	0	0	1

Some of the refused connections were legitimate requests, such as the connections to the Ident Protocol on TCP port 113. Attempts to connect to NNTP, Gopher and alternate WWW ports (ports 119, 70, 8001 and 8080) were probably done by people who thought that (or wanted to find out if) I ran these services on Minnie. Most of the remaining TCP connect attempts are probably suspicious. The attempts on bootserver (port 2016), cisco-tna (port 131), rlogin (port 513) and the snmp-tcp-port (port 1993) are definitely suspicious.

## UDP Port Attempts for June – December 1995

Port	Service	Jun	Jul	Aug	Sep	Oct	Nov	Dec	Sum
512	biff	9	18	9	8	8	88	48	188
1525	prospero	2	3	0	0	22	10	0	37
518	ntalk	0	6	2	0	0	0	0	8
517	talk	0	7	0	0	0	0	0	7
161	snmp	0	0	0	0	0	3	0	3
8305		0	0	0	2	0	0	0	2
525	timed	0	0	0	1	0	0	0	1
1874		0	1	0	0	0	0	0	1
62001		0	0	0	0	0	0	1	1
62010		0	0	0	0	0	0	1	1
62015		0	0	0	0	0	0	1	1

I have removed broadcast packets (mainly to UDP port 520) and packets from the `traceroute` command. The attempts to connect to a Prospero server on UDP port 1525 again were probably from people who thought I ran Prospero. The attempts to connect to `talk` and `ntalk` (ports 517 and 518) are dubious. I would view any attempts to connect to SNMP (port 161) as suspicious.

Minnie does not attract much attention as she is a small server on the Internet. Results from machines like `archie.au` or `munnari.oz.au` would be much more interesting. Surprisingly, there were no attempts to connect to TFTP, NFS, X11, `rexec` or the Sun RPC port.

## Packet Suckers

It's nice to know which ports are attracting attention, but it would be good to log the packets on those ports where suspicious activity occurs. In [Bellovin 92], Steven Bellovin describes the use of programs to log TCP and UDP activity, and a program called `portmopper` to monitor Sun RPC requests. Unfortunately, he hasn't made these programs publicly available, despite their usefulness.

While drafting this paper, I decided to write my own replacements. I now have `tcpsuck` and `udpsuck`, two programs which are run by `inetd` to log TCP and UDP packets on particular unused ports. For ports which do have services on them, TCP Wrappers can divert dubious requests to these programs, while still servicing legitimate users.

Because these programs were only recently written, none have yet been triggered by would-be intruders. However, here are some example logs where I have tried to break in to Minnie from a non-ADFA site.

- An attempt to TFTP my `/etc/passwd` file:

```
Jan 25 12:15:10 Pkt from some.where.com port 1062 to tftp
0- 00012f65 74632f70 61737377 64006e65    ../etc/passwd.ne
16- 74617363 696900                      tascii.
```

- A root rlogin attempt:

```
Jan 25 13:35:03 Data from some.where.com port 1016 to login
0- 00776b74 00726f6f 74007874 65726d2f    .wkt.root.xterm/
16- 39363030 00                      9600.
```

- An NFS mount request:

```
Jan 25 13:37:48 Pkt from some.where.com port 836 to sunrpc
0- 320a406e 00000000 00000002 000186a0    2.0n.....
16- 00000002 00000003 00000000 00000000    .....
32- 00000000 00000000 000186a3 00000002    .....
48- 00000011 00000000    .....
```

Due to the design of the Sun RPC protocols, a simple UDP packet logger won't capture the attempted RPC request. A program like Bellovin's portmopper program is required; I am working with Wietse Venema to construct such a program.

## Work in Progress

There are still many other things in the TCP/IP network stack and in the network server programs to fix. I am concentrating on the kernel side of things. The following work is only a few weeks old, and needs a lot of in-field testing before I can claim that it is robust and safe.

Nearly all TCP/IP network stacks derived from 4BSD have a glaring hole in the generation of TCP sequence numbers. This can be used to fake a connection to a computer from what it thinks is a trusted host. The hole has been noted in [Morris 85] and [Bellovin 89] and has been successfully used to break in to many machines. Read these papers and the recent CERT advisories for more information. I have a fix for the TCP sequence number hole, but I want it checked by some knowledgeable beta-testers before I release it on an unsuspecting Internet.

Other such attacks aimed at the TCP/IP network stack involve advertising a forged route to a system, either using ICMP redirects or IP packets with the Source Routing option. Most of these can be foiled by placing appropriate packet filtering on the routers that connect you to the Internet. A few added lines in the FreeBSD kernel will log these sorts of packets, and ignore the advertised routes.

## Conclusion

There is a ongoing 'arms race' between network intruders and network system administrators. Both are developing new tools to defeat the other, but in many cases the system administrator tools can be used by would-be network intruders to find network vulnerabilities and exploit them. Programs like TCP\_Wrappers perform very useful work by allowing or refusing connections based on the requester's IP address, but do not log successful and unsuccessful connections (or packet contents) on unused TCP and UDP ports. Many systems are also vulnerable to spoofing of the IP and TCP protocols via various mechanisms.

I have added code to the FreeBSD kernel to detect connections on unused TCP and UDP ports. Results to date show little suspicious behaviour. Recently, I have written programs to log actual packet contents for connection attempts on suspicious ports, which should give even more information about attempted breakins. A similar program to log Sun RPC attempts is still required.

There are many other suspicious network events which could be logged, such as packets with strange options, protocol spoofing and routing redirections, and I am working on some more kernel modifications to log and prevent attacks of this nature.

Logging information is useless unless the information it contains is brought to the system administrator's attention in a useful and timely way. Automated and semi-automated tools are needed to summarise the logged data and present it to the administrator in a way which highlights activity which she/he considers suspicious.

Finally, individual system administrators can only monitor their own systems. They are not in a position to perceive trends in intruder activity. We are fortunate in Australia to have an organisation like AUSCERT which can deal with intruder activity and provide help to the Internet community. The collection of suspicious activity logs from several machines by an organisation such as AUSCERT may help to detect potential intruders before they wreak too much havoc. This can only be done if the information can be provided in a rational and machine-parseable format, and if the collection and analysis is performed in a (nearly) fully-automatic manner.

## Acknowledgments

The people at AUSCERT, especially Danny Smith and Eric Halil, gave me a lot of useful feedback in the preparation of this paper, and suggested changes to my kernel modifications to further improve network monitoring. Lawrie Brown at ADFA also helped by proofing the early drafts and providing constructive comments.

## Appendix – References and Where to Get It?

One of the best places in Australia to get information about network security is from AUSCERT's Web server <http://www.auscert.org.au/> and anonymous FTP server <ftp://ftp.auscert.org.au/>. You will find advisories about security vulnerabilities from both AUSCERT and CERT, information about how to improve your security, and tools to help you improve and monitor your security. You will find SATAN, Strobe and the TCP\_Wrappers on the FTP server, along with other tools like COPS. Most of the papers in the bibliography should be on the FTP site too.

If you have control over your network connection, and in particular the routers, you should consider implementing some form of firewall. There are several commercial and freeware products to help you do this. Before you start, get one of the following two books:

**Building Internet Firewalls**, Brent Chapman & Elizabeth Zwicky. Published by O'Reilly & Associates, Inc. ISBN 1-56592-124-0.

**Firewalls and Internet Security**, William Cheswick & Steven Bellovin. Published by Addison-Wesley. ISBN 0-201-63357-4.

These books outline what a firewall is, what it can and *cannot* do, how to set up firewalls, routers and bastion hosts, how to construct a network security policy and what to do if you are attacked. Cheswick & Bellovin's book was reviewed in the February 1995 AUUGN, and Chapman & Zwicky's book was reviewed in the October 1995 AUUGN.

The kernel patches and user-mode tools I've described in this paper are available at <ftp://minnie.cs.adfa.oz.au/pub/NetSecurity/>. In there you will find the following tar files:

**kernmods.tar.gz**: These are the FreeBSD kernel modifications to log connections to unused files, log root execs and log/prevent IP spoofing. Apply these to your kernel at your own risk!

**pktsuckers.tar.gz**: These are user-mode programs to log the contents of packets to suspicious ports. To get the most out of these programs, you should get the TCP\_Wrappers from AUSCERT's FTP server.

**wktportmopper.tar.gz**: This is a replacement portmapper daemon which will log unauthorised RPC requests and also log the actual requests themselves. You will need the TCP\_Wrappers to use this program.

## References

[Bellovin 89] S. M. Bellovin. *Security Problems in the TCP/IP Protocol Suite*. *Computer Communication Review*, 19(2):32–48, April 1989.

[Bellovin 92] S. Bellovin. *There Be Dragons*. In *Proceedings of the 3rd Usenix UNIX Security Symposium*, 1992.

[Bellovin 93] S. M. Bellovin. *Packets Found on an Internet*. *Computer Communication Review*, 23(3):26–31, July 1993.

[Morris 85] R. T. Morris. *A Weakness in the 4.2BSD Unix TCP/IP Software*, 1985. Ftp'd from [research.att.com](http://research.att.com).

- mk, not make
- compilers, assemblers
- sam—split between display and editing
- no uucp

### SIGNIFICANT POINTS OF PLAN 9

- file systems for many resources
- private name spaces

### DISCUSSION

- conventional monolithic kernel
- 9P not extensible
- suggest replacing streams with static queues
- fs code is distinct, requiring too much duplication of drivers
- can't describe namespace—cpu reinterprets lib/profile

### ADFA

- installed 1993 version on standalone PC
- installed 1995 floppy version on networked PC
- installed u9fs with complete readonly CD-ROM images
- modified u9fs for readonly use, for unknown user, for user invocation
- new u9fsstart
- readonly installation of floppy system for general use
- floppy boot
- installing fs on SS2, cpu/auth on SS1+

### PLANS

- get TCP based auth server
- u9fs to do authentication
- dns file system
- get other staff interested and using it
- use in teaching

### IMPRESSIVE BITS

- user interface is acme—mouse chords instead of typing—a nice user interface (also wily)
- alef
- 8½ is a nice window system
  - much less frills than X and xterm—no icons
- rc is a nice shell (I've been using a reimplementations of it on Unix for a number of years as my preferred shell)

- much simpler quoting
- control structures more like C
- sam is a nice editor (I used it for a while before buckling under with dumb VT100 access)

- regular expressions may include parts of several lines
- more regular command language
- REs over file names
- same REs used everywhere in Plan 9 except for rc filename globbing

- debugger is acid—language, not just a command set
- demonstrate 8½, rc, sam, acme
- tools hosted on Unix—sam(term), u9fs, ilgate, rc, wily, 9term, 9wm, libXg
- clean, small OS
- most play is in writing device drivers or reimplementing typical tasks using new tools (news reader)
- good fun
- nice to use

### HOW DO YOU GET IT

- most sources are supplied—not crypt, ksh, or C++
- ISBN 0-03-017143-1 (USD350) for 1 CD-ROM, 4 floppies, 2 books
- ISBN 0-03-017142-3 (USD125) for books only
- book 1 is manuals, book 2 is other documents, many (but not all) of which are available elsewhere
- you could try H-B in Sydney—when I did, they didn't know about it
- Harcourt-Brace is +1 407 345 3800 (somewhere near Orlando, Florida, timezone EST/EDT)
- may need to tell them you're in Australia, so they can check whether their local distributor is supposed to have it
- choice of FedEx, airmail, slow
- shrink-wrap site licence for non-commercial use, AT&T will negotiate for commercial use

### HELP

- WWW page <http://plan9.att.com/plan9/>
- mailing list 9fans@cse.psu.edu
- newsgroup comp.os.plan9

# The PDP-11 Unix Preservation Society or What's So Great About Unix, Anyway?

Warren Toomey  
wkt@cs.adfa.oz.au

School of Computer Science  
Australian Defence Force Academy

## Abstract

*Unix as a working operating system is now well over 25 years old. So why hasn't it died out yet? The command line interface is horrible! This talk looks at some of those things that makes Unix good, and shows that nearly all of them were there in the original versions of Unix. These 'vintage' versions of Unix still fascinate some people, which is why a group has been formed to try and preserve them. With luck and a working PDP-11 emulator, a demonstration of Seventh Edition Unix (17 years old) will be given.*

*This paper has been taken from the slides presented at the Summer Conference.*

## A History of Early Unix

Bell Laboratories (BTL) was one of the co-developers of Multics in the late '60s but pulled out in 1969, leaving many BTL people at a loose end. Several people who were still interested in OS design 'found' a PDP-7 and tinkered with ideas, many from their Multics experience. The PDP-7 was slow & painful, and the group managed to get a PDP-11/20 on the promise of building a document processing system, not an OS. `roff` and `ed` were quickly written, and the first Unix system 'shipped' to Bell's Patent Department.

1st Edition Unix was written entirely in assembler, was multiuser (2 users) & multitasking, and had `ar`, `as`, `cat`, `cp`, `df`, `du`, `ln`, `ls`, `mail`, `od`, `su`, `wc` and `who`. By 2nd Edition there were 10 Unix installations; 3rd Edition had 16. 3rd & 4th Edition saw the system rewritten almost entirely in C and the introduction of pipes.

Doug McIlroy had been nagging Ken Thompson about implementing pipes. Finally Ken hacked the system to do it, and then hacked the user programs to use 'standard input'. "It was clear, practically minutes after the system came up with pipes working, that it was wonderful... Nobody would ever go back and give that up ..." – Dick Haight. Pipes created the Unix 'toolbox' concept.

## A History of Early Unix

Ken Thompson's paper about 4th Edition Unix in the July '74 CACM caused a stir around the academic world. Many institutions asked for copies of Unix. AT&T at the time could not sell software, so they licensed copies of Unix and provided no support or bug fixes. Thus Unix spread around the world and users were forced to fend for themselves. As they had the source, they fixed bugs, developed new programs and kernel modifications, and came to depend on each other for support.

Throughout the '70s, Unix predominately ran on PDP-11s, although the use of C meant that the system could be ported to other machines. User communities such as AUUG, EUUG and USENIX sprang up. BTL kept developing Unix, incorporating many changes from the users. Within AT&T, there were many 'flavours' of Unix (MERT, Programmer's Work Bench), and many developments were folded back into the research versions.

The last research version before the AT&T commercial versions was 7th Edition, considered by many to be the 'one true Unix'. 7th Edition was ported to the VAX (as 32V), which became the predominant Unix platform of the early- to mid-80s. AT&T were allowed to sell software in 1983, and Unix was sold as 'System V' and its offspring.

## What's So Great About Unix?

By 7th Edition & 32V, most of the 'great' features of Unix were in place.

**Concepts – The Toolbox approach:** Write programs that do one thing & do it well. Write programs to work together. Write programs to handle text streams.

**Concepts – Device & Machine Independence.**

**Source Code:** The kernel source was compact and elegant. Full source code for the kernel and applications was available to the users, making it readable and fixable. The source code was written in a high-level language, allowing system portability.

**Features:** Pipes. A small set of system calls. Multitasking, multiuser, file protections. Virtual memory in 32V. An advanced file system design.

**Communal Spirit:** User groups such as AUUG and USENIX. Development of the Usenet with UUCP. Free Software Foundation as a reaction to commercial Unix. The Internet through the development of BSD 4.x. This communal spirit now permeates the Linux and Free BSD user groups.

## AUUG's Development in Unix Development

Many of the changes done to 6th & 7th Edition Unix were done by Australians, and AUUG members in particular. Several people from UNSW hacked 6th Edition to perform user accounting and resource sharing; the result was the 'Australian Unix Share Accounting Method' or AUSAM.

Robert Elz at the University of Melbourne gave Unix quotas and autoconfiguration, both still in current versions of Unix. He also did a lot of work on the terminal drivers.

The University of Sydney developed the 'Sydney Uni Network' software, which became the basis of the ACSnet, before the Internet arrived.

John Lions from UNSW (with help from students) created his famous 'Unix Commentary' which described the 6th Edition Unix kernel line by line, and was used to teach OS concepts to students.

## The PDP Unix Preservation Society

Some people collect old cars; some people collect antiques. The PDP Unix Preservation Society is devoted to the preservation of all information related to the versions of Unix that ran on PDPs.

The Web page <http://minnie.cs.adfa.oz.au/PUPS/> has the stuff that can be obtained licence free, such as anecdotes, some documentation, images etc. A protected ftp archive has the stuff that needs a Unix license. Currently we have 5th, 6th & 7th Edition, PWB with AUSAM mods, 2.9BSD and 2.11BSD.

There is a mailing list of 20 people for those who are still using PDP Unixes, or who like to reminisce. The latest BSD Unix for the PDP, 2.11 BSD, is still being developed by Steven Schultz, and has many of the 4.4BSD-Lite features.

The biggest problem for the group is who to ask for new licenses for 7th Edition Unix. Not AT&T, not Novell, perhaps HP and SCO? The PDP Unix Preservation Society will be more than happy to copy information on paper, mag tapes and anecdotes from you.

## A Look at 7th Edition Unix

7th Edition ran on PDP 11/40s, 11/45s and 11/70s. A typical system might have been:

- An 11/45 with 256K of main memory and a few 10M RL02 drives.
- A dozen or more serial ports connected to terminals (ADM 3a, VT52 or Telerays). Top speed was 9600 baud.
- A 1600bpi 9-track tape drive.
- A lineprinter.
- Maybe a 9600 baud Trailblazer modem for ACSnet access.

There were several languages: cc, f77, awk, sed, lex, yacc. Document processing: troff, nroff, pic, eqn, tbl. The editor was ed – no vi or emacs yet. UUCP was the only networking code; no TCP/IP.

The kernel was around 50K in size. ls was 8K. troff was 48K. The operating system was around 17,000 lines of code, including header files. It fitted into 310K of disk space.

## References and Further Reading

*A Quarter Century of UNIX* by Peter Salus is the book to read for a historical look at Unix, its ethos and its communal spirit. *Life with UNIX* by Don Libes & Sandy Ressler gives a briefer look at the history and development of Unix. Old newsletters from AUUG, EUUG and Usenix are fascinating. Will AUUG ever set up some form of library/archive? 'UNIX Review' had several interviews with the early Unix developers in 1985.

The manuals from the 6th & 7th Editions are succinct, with humour not found in newer Unix versions. They also have several tutorials and papers, including the 1974 CACM paper. If you can get it, John Lions *Commentary on the UNIX OS* is a great read. John Lions & Peter Salus are working on getting this published properly.

## Conclusion

Many users regard an operating system's user interface as the operating system itself. This view is increasing as users become GUI-dependent. What made Unix so great was not the user interface. Nor was it the implementation (7th Edition, System V, Solaris etc.).

Unix is good because it embodies a set of powerful concepts in a flexible environment. Users could change their user-interface. Programmers had access to a small but sophisticated set of OS APIs. The APIs were *independent* of the hardware. The system could be *changed* if the users desired it. By writing code, users learned of the system's limitations and the assumptions made by the designers.

David Tilbrook wrote: "[Unix] wasn't a great advance in computing; if anything it was a great simplification. [It gave] the user things that were inconceivable prior to that."

Since 7th Edition, Unix has put on a lot of middle-age spread. What allows it to survive is the core of elegant concepts that it contains.

